

# METHOD OF PUBLIC KEY GENERATION

## FIELD OF THE INVENTION

5 The present invention relates to public key cryptosystems and more particularly to key generation within such systems.

## BACKGROUND OF THE INVENTION

10 The basic structure of a public key cryptosystem is well known and has become ubiquitous with security in data communication systems. Such systems use a private key  $k$  and a corresponding public key  $\alpha^k$  where  $\alpha$  is a generator of the group. Thus one party may encrypt a message  $m$  with the intended recipients public key and the recipient may apply his private key to decrypt it.

15 Similarly, the cryptosystems may be used for key agreement protocols where each party exponentiates the other party's public key with their own private key. Thus party A will take B's public key  $\alpha^b$  and exponentiate it with A's private key  $a$  to obtain a session key  $\alpha^{ab}$ . Similarly, B will take A's public key  $\alpha^a$  and exponentiate it with B's private key  $b$  to obtain the same session key  $\alpha^{ab}$ . Thereafter data may be transferred using a symmetric key protocol utilizing the common  
20 session key.

Public key cryptosystems may also be used to sign messages to authenticate the author and/or the contents. In this case the sender will sign a message using his private key and a recipient can verify the message by applying the public key of the sender. If the received  
25 message and the recovered message correspond then the authenticity is verified.

The public key cryptosystems rely on the intractability of the discrete log problem in finite field arithmetic, that is even when the generator  $\alpha$  and public key is known, it is computationally infeasible to obtain the corresponding private key. The security of such systems  
30 does therefore depend on the private key remaining secret. To mitigate the opportunity of disclosing the private key, protocols have been developed that use a pair of private keys and

corresponding public keys, referred to as long term and short term or ephemeral key pairs respectively. The ephemeral private key is generated at the start of each session between a pair of correspondents, usually by a random number generator. The corresponding ephemeral public key is generated and the resultant key pair used in one of the possible operations described above.

- 5 The long-term public key is utilized to authenticate the correspondent through an appropriate protocol. Once the session is terminated, the ephemeral key is securely discarded and a new ephemeral key generated for a new session.

10 Some of the more popular protocols for signature are the ElGamal family of signature schemes such as the Digital Signature Algorithm or DSA. The DSA algorithm utilizes both long term and ephemeral keys to generate a signature of the message. The DSA domain parameters are preselected. They consist of a prime number  $p$  of a predetermined length, by way of example 1024 bits; a prime number  $q$  of a predetermined bit length, by way of example 160 bits, where  $q$  divides  $p-1$ ; a generator  $\alpha$  lying between 2 and  $p-1$  and which satisfies the condition  
15  $(\alpha^q \bmod p) = 1$ , and; a cryptographic hash function  $H$ , such as SHA-1.

The DSA requires the signatory to select an ephemeral key  $k$  lying between 1 and  $q-1$ . A first signature component  $r$  is generated from the generator  $\alpha$  such that  $r = (\alpha^k \bmod p) \bmod q$ . A second signature component  $s$  is generated such that  $s = k^{-1}(H(m) + dr) \bmod q$ , and  $d$  is the long term private key of the signatory. The signature on the message  $m$  is  $(r,s)$ . The signature may be  
20 verified by computing

$$H(m),$$

$$u_1 = s^{-1}H(m) \bmod q$$

$$u_2 = s^{-1}r \bmod q$$

25  $v = \alpha^{u_1} \beta^{u_2} \bmod p$ , where  $\beta = \alpha^d \bmod p$  is the long term public key of the signatory and finally verifying that  $r = v \bmod q$ . The use of both the ephemeral and long-term keys in the signature binds the identity of the signatory to the ephemeral key but does not render the long-term key vulnerable.

A similar signature protocol known as ECDSA may be used for elliptic curve  
30 cryptosystems. In this protocol  $k$  is selected in the interval 1 to  $n-1$  where  $n$  is an  $l$  bit prime. The signature component  $r$  is generated by converting the  $x$  coordinate of the public key  $kP$ , where  $P$

is the seed point on the curve, to an integer mod  $n$ , i.e.  $r = x_{kP} \bmod n$ . The component  $s = k^{-1}(H(m) + dr) \bmod n$  and the signature on the message  $m$  is  $(r, s)$ .

It will be apparent in ElGamal signature schemes such as the DSA and ECDSA, that if an ephemeral key  $k$  and the associated message  $m$  and signature  $(r, s)$  is obtained it may be used to yield the long term private key  $d$  and thereafter each of the ephemeral keys  $k$  can be obtained. Neither the DSA nor the ECDSA inherently disclose any information about the public key  $k$ . They both require the selection of  $k$  to be performed by a random number generator and it will therefore have a uniform distribution throughout the defined interval. However the implementation of the DSA may be done in such a way as to inadvertently introduce a bias in to the selection of  $k$ . This small bias may be exploited to extract a value of the private key  $d$  and thereafter render the security of the system vulnerable. One such implementation is the DSS mandated by the National Institute of Standards and Technology (NIST) FIPS 186-2 Standard. The DSS stipulates the manner in which an integer is to be selected for use as a private key. A seed value,  $SV$ , is generated from a random number generator which is then hashed by a SHA-1 hash function to yield a bit string of predetermined length, typically 160 bits. The bit string represents an integer between 0 and  $2^{160} - 1$ . However this integer could be greater than the prime  $q$  and so the DSS requires the reduction of the integer mod  $q$ , i.e.  $k = \text{SHA-1}(\text{seed}) \bmod q$ .

Accordingly the algorithm for selecting  $k$  may be expressed as :-

if  $\text{SHA-1}(\text{seed}) \geq q$  then  $k \leftarrow \text{SHA-1}(\text{seed}) - q$   
else  $k \leftarrow \text{SHA-1}(\text{seed})$ .

With this algorithm it is to be expected that more values will lie in the first interval than the second and therefore there is a potential bias in the selection of  $k$ .

Recent work by Daniel Bleichenbacher suggests that the modular reduction to obtain  $k$  introduces sufficient bias in to the selection of  $k$  that an examination of  $2^{22}$  signatures could yield the private key  $d$  in  $2^{64}$  steps using  $2^{40}$  memory units. This suggests that there is a need for the careful selection of the ephemeral key  $k$ .

## SUMMARY OF THE INVENTION

It is therefore an object of the present invention to obviate or mitigate the above disadvantages in the generation of a private key.

5

In general terms the present invention provides a key generation technique in which any bias is eliminated during the selection of the key.

## BRIEF DESCRIPTION OF THE DRAWINGS

10

Embodiments of the invention will now be described by way of example only with reference to the accompanying drawings in which:-

Figure 1 is a schematic representation of a data communication system;

Figure 2 is a flow chart showing a first embodiment of key generation;

Figure 3 is a flow chart showing a second embodiment;

Figure 4 is a flow chart showing a third embodiment;

Figure 5 is a flow chart showing a fourth embodiment;

Figure 6 is a flow chart showing a fifth embodiment; and

Figure 7 is a flow chart showing a sixth embodiment.

## DESCRIPTION OF THE PREFERRED EMBODIMENTS

30

Referring, therefore to figure 1, a data communication system 10 includes a pair of correspondents 12, 14 connected by a communication link 16. The link 16 may be a dedicated link, a multipurpose link such as a telephone connection or a wireless link depending on the particular applications. Similarly, the correspondents 12, 14 may be computer terminals, point-of-sale devices, automated teller machines, constrained devices such as PDA's, cellphones, pagers or any other device enabled for communication over a link 16.

35

Each of the correspondents 12, 14 includes a secure cryptographic function 20 including a secure memory 22, an arithmetic processor 24 for performing finite field operations, a random number generator 26 and a cryptographic hash function 28 for performing a secure cryptographic hash such as SHA-1. The output of the function 28 will be a bit string of predetermined length, typically 160 bits although other lengths such as 256, 384 or 512 are being used more frequently. It will be appreciated that each of these functions is controlled by a processor executing instructions to provide functionality and inter-operability as is well known in the art.

The secure memory 22 includes a register 30 for storing a long-term private key,  $d$ , and a register 32 for storing an ephemeral private key  $k$ . The contents of the registers 30, 32 may be retrieved for use by the processor 24 for performing signatures, key exchange and key transport functions in accordance with the particular protocols to be executed under control of the processor.

The long term private key,  $d$ , is generated and embedded at the time of manufacture or initialization of the cryptographic function and has a corresponding long-term public key  $\alpha^d$ . The long-term public key  $\alpha^d$  is stored in the memory 22 and is generally made available to other correspondents of the system 10.

The ephemeral key,  $k$ , is generated at each signature or other cryptographic exchange by one of the routines disclosed below with reference to figures 2 to 9. Once the key,  $k$ , and corresponding public key  $\alpha^k$  is generated, it is stored in the register 32 for use in the cryptographic protocol, such as the DSA or ECDSA described above.

Referring, therefore, to figure 2, a first method of generating a key,  $k$ , originates by obtaining a seed value (SV) from the random number generator 26. For the purposes of an example, it will be assumed that the cryptographic function is performed over a group of order  $q$ , where  $q$  is a prime represented as a bit string of predetermined length  $l$ . By way of example only it will be assumed that the length  $l$  is 160 bits, although, of course, other orders of the field may be used.

To provide a value of k of the appropriate order, the hash function 28 has an l bit output, e.g. a 160 bit output. The bit string generated by the random number generator 26 is greater than l bits and is therefore hashed by the function 28 to produce an output H(seed) of l bits.

5

The resultant output H(seed) is tested against the value of q and a decision made based on the relative values. If  $H(\text{seed}) < q$  then it is accepted for use as k. If not, the value is rejected and the random number generator is conditioned to generate a new value which is again hashed by the function 28 and tested. This loop continues until a satisfactory value is obtained.

10

A further embodiment is shown in figure 3. In this embodiment, the output of the random number generator 26 is hashed by hash function 28 as before and tested against the value of q. If the H(seed) value is not accepted, the output of the random number generator 26 is incremented by a deterministic function and rehashed by function 28.

The resultant value H(seed) is again tested and the procedure repeated until a satisfactory value of k is obtained.

The output may be incremented by adding a particular value to the seed value at each iteration, or may be incremented by applying a non-linear deterministic function to the seed value. For example, the output may be incremented by applying the function  $f(\text{seed}) = a \cdot \text{seed}^2 + b \bmod 2^{160}$ , where a and b are integer constants.

25 A further embodiment is shown in figure 4 which has particular applicability to an elliptic curve cryptosystem. By way of example it will be assumed that a 163 bit string is required and that the output of the hash function 28 is 160 bits.

30 The random number generator 26 generates a seed value SV which is processed by the hash function 28 to obtain a first output H(seed).

The seed value SV is incremented by a selected function to provide a seed value SV+ which is further processed by the hash function 28 to provide a second output H(seed+).

5 The two outputs are then combined, typically by cocatenation, to produce a 320 bit string H(seed)//H(seed+). The excess bits, in this case 157 are rejected and the resultant value tested against the value of q. If the resultant value is less than q, it is accepted as the key k, if not the value is rejected.

10 Upon rejection, the random number generator may generate a new value as disclosed in figure 2 or may increment the seed value as disclosed in figure 3.

A further embodiment is shown in figure 5 which is similar to that of figure 4. In the embodiment of figure 5, the selection of the required l bit string is obtained by applying a l-bit wide masking window to the combined bit string.

This is tested against the value of q and if acceptable is used as the value of k. If it is not acceptable it is rejected and the l bit window incremented along the combined bit string to obtain a new value.

20 The values are tested and the window incremented until a satisfactory value is obtained.

A similar procedure may be used directly on an extended output of the hash function 28 as shown in figure 6 by applying a window to obtain the required l bit string. The bit string is tested against q and the window incremented until a satisfactory value of k is obtained.

25 As shown in figure 7, the value of k may be generated by utilizing a low Hamming weight integer obtained by combing the output of the random number generator 26 to facilitate computation of an intermediate public key  $\alpha^k$ . The integer is masked by combination with predetermined precomputed value k' to obtain the requisite Hamming weight for security. Such a procedure is disclosed in copending Canadian application 2,217,925. This procedure is modified to generate the low Hamming weight integer k as a bit string greater than l, for

example, a 180 bit string. The masking value  $k'$  is distributed throughout the 180 bit string and the resultant value reduced mod  $q$  to obtain a 163 bit value  $k''$ . Note that the value  $\alpha^{k''}$  can be efficiently computed by combining the precomputed value  $\alpha^{k'}$  with the efficiently computable value  $\alpha^k$ .

5

A similar technique may be used by relying on multiplicative masking. In this embodiment the value of  $k$  is combined with a value  $\beta$  where  $\beta = \alpha^u$ . The value of  $u$  is a secret value that is used to mask the low Hamming weight of  $k$ . Again, the values of  $u$  and the low Hamming weight number  $k$  can be chosen to have bit lengths greater than  $l$ , for example, bit lengths of 180. The resultant value is  $k'' = u^k \text{ mod } q$ . It will be appreciated that  $\alpha^{k''}$  can be efficiently computed since  $\beta = \alpha^u$  is precomputed, and since  $k$  has low Hamming weight.

10

Although the invention has been described with reference to certain specific embodiments, various modifications thereof will be apparent to those skilled in the art without departing from the spirit and scope of the invention as outlined in the claims appended hereto.

15